Accepted for International Workshop on Issues and Applications of Database Technology (IADT'98), Berlin, Germany, July 6–9, 1998.

TRANSPARENT INTEGRATION OF CONTINUOUS MEDIA SUPPORT INTO A MULTIMEDIA DBMS

Silvia Hollfelder, Florian Schmidt, Matthias Hemmje, Karl Aberer
GMD - German National Research Center for Information Technology
Integrated Publication and Information Systems Institute (IPSI)
Dolivostr. 15, D-64293 Darmstadt, Germany
e-mail: {hollfeld, fschmidt, hemmje, aberer}@darmstadt.gmd.de

ABSTRACT

Multimedia Database Management Systems (MM-DBMS) have to efficiently provide the specific functionalities required by time-dependent multimedia data types. During presentation playout, this requires support for continuous data delivery and media-oriented optimization of presentation quality according to users' requirements. In this work, we describe our way to integrate this type of support transparently with other MM-DBMS functions like media editing and querying. This has been achieved by extending an existing data model, in particular by means of stream abstractions for editing and presentation, and by extending the system architecture by means of supporting stream processing components, e.g., for intra-media synchronization by adaptation. This approach has been implemented as an extension of existing commercial object-relational DBMS technology.

1 INTRODUCTION

In commercial DBMS there is a clear trend to provide specialized functionality, which has usually been offered as separate software packages and required substantial development efforts for the integration with database applications, as integrated database extensions. Examples are particularly found in the management of nonstandard or multimedia data types, e.g., text retrieval, image handling, statistical functions. This development is a key factor for the rapid spreading of object-relational database technology among all major database vendors, and it is the reaction to the requirements of new applications utilizing distributed multimedia technology, like digital video production [S96][SH94], education and training or digital libraries [BAN95]. For static data and media, this approach is relatively straightforward as it does not heavily affect the fundamental processing paradigms used in conventional DBMS. For handling continuous media, like, e.g., video and audio, additional characteristics and requirements related to the presentational aspects of these media need to be considered. Currently, there are two "schools" to that respect (for a nice discussion see also [O96]):

1. Media presentation is not a problem that is to be considered by the DBMS. The DBMS provides a storage framework for the media, including metadata required for search and manipulation. Presentation is supported by external systems, like video servers, or within distributed multimedia applications.

2. The DBMS has to support presentational aspects, since this is a fundamental aspect of accessing the media data managed by the DBMS. Presentational requirements like synchronization are to be considered as a new type of constraints that need consistent support by the DBMS [MS96].

From our perspective there are several arguments in favor of the second school.

1. Presentations of composite multimedia objects require the coordinated scheduling of the available resources based on the characteristics of and relationships between the involved media. Since this information is stored in the database the DBMS itself most efficiently performs the scheduling. For example, when presenting a video with text inserts the DBMS needs to support a buffering scheme for the video that exploits the stream semantics and expected user interactions. In addition, it has to schedule the delivery of the text inserts such that they can be presented timely.

2. In case of concurring requests, the DBMS is the right place to optimize the database usage and usage of available different media, for example by performing quality adaptations [TKCWP97].

3. Different abstractions are used in the different stages of processing media data. For example, an MPEG video stream is edited at the frame granularity, retrieved at the granularity of scenes or shots, and distributed over a network for presentation at the granularity of group-of-

This work has partly been funded by the European Union within the framework of the ESPRIT Long Term Research Project HERMES, No. 9141 [http://www.ced.tuc.gr/hermes/hermes.html].

pictures. The different data structures representing these abstractions, which are also relevant for presentation purposes, need to be kept consistent during updates.

4. Exploiting application-specific characteristics of multimedia objects for distribution and presentation of the media increases the necessity of integrated presentation support in the DBMS. Examples are quality adaptations for parts of presentations resulting from a retrieval request parametrized by their relevance, buffering strategies taking into account possible future interactions based on application semantics, e.g., favorite hyperlinks, or resource scheduling taking user profiles or usage statistics into account.

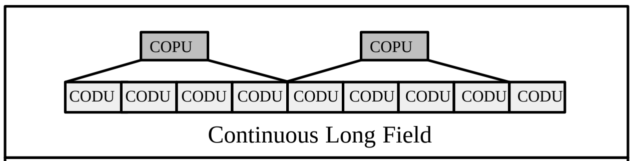
In this paper, we show the feasibility of the integration of presentational support for multimedia data with a standard commercial object-relational DBMS (Informix Universal Server IUS). Basic support in the spirit of school 1 is provided by the Informix Universal Server by means of the Video Foundation Blade [VFUG97]. We actually take advantage of the mechanisms provided by this database extension to simplify the physical integration of the media server and obtain additional abstraction and querying capabilities. The additional presentational support that we provide addresses the continuous data transport required during the presentation of continuous media streams, client-side buffering mechanisms exploiting stream and interaction semantics, and client-side quality adaptation mechanisms to overcome fluctuations in the network and server resources. Most of the concepts were already developed when we extended the object-oriented DBMS VODAK [V95] to a (proprietary) MM-DBMS [RKN96]. This paper shows that the concepts for presentational support, developed in this earlier proprietary work, can be smoothly integrated into standard platforms, taking advantage of the available functionality of these systems. Thus there is no principal difficulty in a modular extension of modern, extensible DBMS with presentational support for multimedia data.

2 GENERAL CONCEPTS

In the following, we go into some details of the basic concepts supporting the storage and presentation that have been integrated as multimedia database extensions to give presentational support to multimedia objects.

2.1 CONTINUOUS LONG FIELDS ABSTRACTIONS

We provide a data type *Continuous Long Field* (CLF) as a generic representation for any kind of continuous media, like audio, video or animations. This datatype supports operations for editing and presentation, like "insert", "request", or "delete". Thus it combines the concepts of long fields and streams. Structural metadata, like the format and other recording parameters, are stored with every CLF object. Different media encoded in a single stream (like, e.g., with MPEG encoding) are modelled as separate CLF objects, to enable the individual retrieval and manipulation of each of the media parts. For editing purposes, a CLF object is segemented into a sequence of manipulation units, called Continuous Object Data Units (CODU). The granularity of these units is determined individually for each type of media, e.g. Motion-JPEG frames, MPEG-1 frames or audio samples. At this level of granularity, it is possible to manipulate the data stream, e.g., by inserting, deleting or appending CODUs.



Media Type	**Audio**	**M-JPEG**	**MPEG**
COPU	Block of samples	Frame	Group of Pictures
CODU	Sample	Frame	Frame

Fig. 1 : Modelling of Continuous Long Fields

For continuous presentation, our approach supports a client-pull mechanisms, i.e., the client continuously requests chunks of data which are provided by the server in a best-effort manner. Since requests in the granularity of CODUs are not very effective and presentation engines, like, e.g., MPEG players or other video and audio devices, handle larger units of data, a second abstraction is introduced to support presentation. An atomic unit requested during synchronous presentation is called a Continuous Object Presentation Unit (COPU). One COPU may consist of several CODUs as illustrated in figure 1. Furthermore, the figure shows that a continuous long field consists of several COPU and CODU objects.

The CLF data type that provides the above abstractions has been implemented as part of our database extension for basic continuous media support. It provides, in particular, storage support for the raw media data and associated metadata, like, e.g., content descriptions, physical characteristics or presentational characteristics. More details on the implementational aspects will be provided in section 3.

2.2 BUFFER MECHANISM

An intelligent buffering mechanism for continuous media streams not only has to optimize access to physical storage it also has to take into account the presentational requirements of the media stream. We use an algorithm called Least/Most Relevant for Presentation (L/MRP) [MKK95] for preloading and replacing of COPUs in the client buffer. Its goal is to support a synchronized presentation of the media stream.

The algorithm considers the current presentation state and likely interactions when deciding which COPUs need to be preloaded and replaced in the client buffer. For that purpose, it assigns a relevance value to each COPU. COPUs with least relevance are replaced first and those with highest relevance value are preloaded first. The relevance value represents the possibility that a COPU may be required in the future. Hereby it is considered that user interactions change the presentation process. Relevance values are distinguished into dynamic and static ones. Dynamic relevance values are derived from the presentation point and the relevance of the

COPU for the progression of the presentation. For example, the COPUs following the presentation point in the current presentation direction have the highest dynamic relevance which decreases with increasing distance to the presentation point. COPUs already presented can still have a fairly high relevance to make them available for change of direction or fast rewind user interactions. Static relevance values are assigned independent of the current presentation point, for example, to working points within the streams that are more likely to be referenced. The functions assigning relevance values can be modelled according to the typical needs of the application.

2.3 BUFFER-TRIGGERED ADAPTATION

Since we do not assume the availability of reservation mechanisms for stream delivery, bottlenecks may occur during presentation. Therefore, we use a client-based adaptation mechanism [DHH93] to dynamically change the presentation quality in order to reduce the data volume that needs to be delivered from the server to the client. In this way, intra-media synchronization can be maintained by reducing disk utilization, memory consumption or required network bandwidth.

Our smooth adaptation technique assigns different presentation qualities to different intervals of client-buffer fill levels. The rationale of this approach is that with high buffer utilization the danger of a bottleneck is lower, while with a lower buffer utilization the danger of loosing intra-media synchronization increases, and the buffer needs to be filled up faster with lower quality data. The intervals used in the adaptation specification overlap. Thus, if the buffer fill level is close to the border of an interval, frequent quality switches are avoided by using hysteresis [AFK95]. The quality adaptations are modelled separately for each media. This enables a transformation of QoS (Quality of Service) parameters, specified by the user, to the adaptation intervals. Adaptation can occur along the temporal dimension (e.g., frame dropping) or along the spatial dimension (e.g., reduction of resolution). In case of dropping, additional constraints can be introduced by specifying the minimum distance in between dropped frames to, e.g., reduce jitter. A detailed description of the adaptation framework is given in [HKR97]. Figure 2 illustrates the architecture of the adaptation feedback and its interaction with the buffering mechanism described in section 2.2. The presentation process starts with a filled client buffer. The presentation engine continuously requests COPUs for their synchronized presentation and handles user interactions. While the COPUs are presented, the preloader asynchronously prefetches COPUs from the server in the client buffer. The adaptation control module controls the buffer size, and, based on the adaptation specification, determines which quality has to be requested. If server CPU, server disk or networks have temporary bottlenecks, the client's buffer utilization starts to decrease. In this case the presentation quality is gracefully degraded. In case of dropping the request of COPUs, logically empty COPUs are inserted into the buffer to increase buffer utilization.

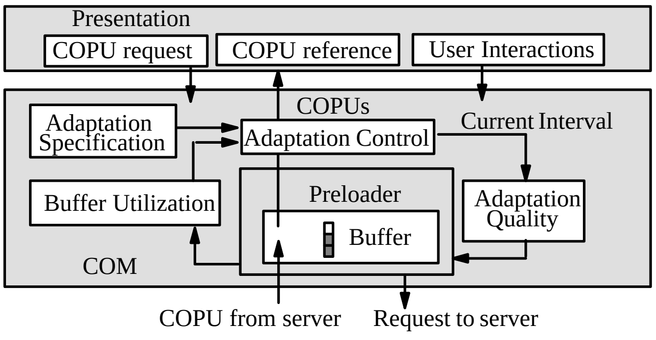


Fig. 2 : Adaptation Feedback Mechanism

The adaptation mechanism has been implemented as part of a continuous media presentation environment of the database client. It constitutes a further component in our database extension for basic continuous media types by supporting intra-media synchronization. Another component, the Reactive Adaptive Playout Manager [RAPM], is responsible for inter-media synchronization and scheduling of composite presentations. It is built on top of the client buffer in our MM-DBMS [TK96].

3 INTEGRATION OF MULTIMEDIA DATA SUPPORT

3.1 OBJECT-RELATIONAL DBMS

Object-relational DBMS are designed to combine the advantages of industrial-strength relational DBMS with the extensibility and expressive powers of object-oriented DBMS. In spite of their high flexibility most object-oriented DBMS lack important features, like powerful declarative access mechanisms, or suffer from scalability problems [D95]. Though the expressiveness and design of object-oriented data models are usually considered to be better, in particular with regard to constructed data types, for most multimedia applications the ability to introduce new abstract data types in object-relational database management systems appears to suffice. The multimedia-related data types, including the necessary operations on them, are added to the kernel at runtime, and can hence be used in the same way as other system-defined data types, e.g., within SQL commands or the database application programmers' interface. The operations are alternatively performed either on the server in case of data intensive processing or on the client in case of computational intensive processing.

For the implementation of basic continuous-media database extensions described in this paper, we used the Informix Universal Server, which has recently become available. Database extensions for the Informix Universal Server are called DataBlades and bundle the implementations of abstract data types both with support functions for the client and standard SQL application schemas.

3.2 THE VIDEO FOUNDATION DATABLADE

The Informix Video Foundation DataBlade (VF DB) provides a basic framework to integrate continuous-media servers with Informix Universal Server. It provides a standardized schema for the control of external storage managers as well as a common interface for the manipulation of the relevant metadata related to physical properties, content or usage of the media streams. This openess in the architecture of the Video Foundation Blade allowed for its use as a basis for the integration of the presentation-oriented mechanisms described before with the Informix Universal Server. The Video Foundation DataBlade itself explicitely omits any support related to the presentation of media streams.

The software architecture of the Video Foundation DataBlade consists of three main components: The Informix Universal Server, an external storage manager and the application program. The Informix Universal Server stores all metadata. It manages and controls the access to the external storage managers and devices. The external storage managers handle the storage and continuous delivery of the various media streams. Whenever an application wants to access continuous objects, it can access them indirectly through the Informix Universal Server Video Foundation DataBlade schema and its access functions. If the access is time-critical, a direct connection to the responsible storage manager (through an appropriate client-server connection) can be mediated through the Informix Universal Server and the Video Foundation DataBlades metadata, as well. If the application requests only metadata, the Informix Universal Server will serve the request on the basis of the Video Foundation DataBlade schema.

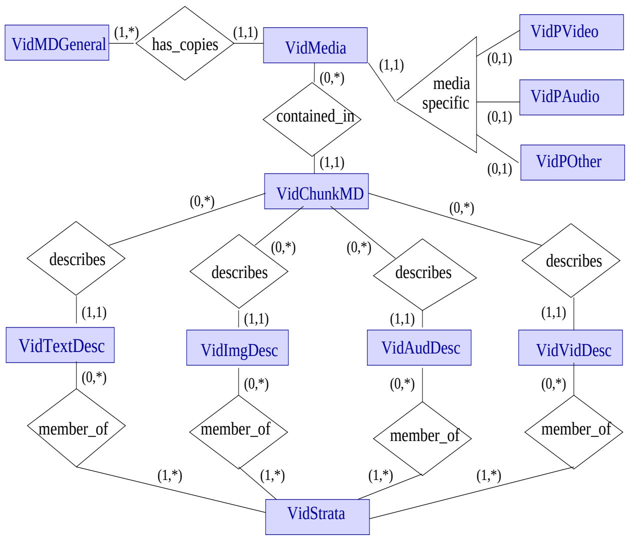


Fig. 3 : Database Schema of the Video Foundation DataBlade

The Video Foundation DataBlade registers all external storage managers within a so-called "MedVSIRegister" table. It also stores representations of objects stored externally as values of the so-called "MedLoc" datatype. To enable a format independent representation of times and periods of time, the Video Foundation DataBlade defines the so-called "MedPtr", "MedTC" and "MedChunk" data types which are responsible for representing exact times in the timing format of the corresponding medium, according to the SMPTE timecode standard.

By means of a "virtual storage interface" (VSI), the Video Foundation DataBlade provides access functions like open, close, read, write, seek, and tell which enable applications to manipulate contents handled by the storage managers. The Video Foundation DataBlade predefines the virtual storage interface for three simple external storage managers: one for files on the client, one for files on the server, and the third for media not available in digitized form.

Besides providing these virtual access and management functions of the virtual storage interface, the Video Foundation DataBlade defines a metadata schema, displayed in figure 3 , which contains tables for technical and physical metadata (stored in so-called "VidMDGeneral", "VidMedia", "VidPVideo", "VidPAudio", "VidPOther", "VidVCodec", "VidACodec", "VidFormat", "VidStandard", "VidPVideo", "VidPAudio", and "VidPOther" tables) as well as for structure- and content-oriented metadata (stored in so-called "VidChunkMD", "VidStrata", "VidVidDesc", "VidImgDesc", "VidAudDesc", and "VidTextDesc" tables). Further information can be found in [VFUG97].

3.3 SYSTEM ARCHITECTURE

In the following, we describe the system architecture that integrates creation, editing, continuous presentation support, and content-based annotation of Continuous Long Fields in the object-relational DBMS Informix Universal Server. Our system is based on a client/server architecture where the server is responsible for the storage of continuous and discrete data. The client is responsible for requesting data from the server, for presenting them and for handling interactions with the user. The architecture consists of four parts, the Informix Universal Server, the Video Foundation DataBlade, the external storage manager and the CLF DataBlade. The CLF DataBlade, in turn, consists of three parts, the DBMS functionality extending the object-relational DBMS, the presentation support module located on the client, and the continuous transport module connecting the presentation support module to the external storage manager during a presentation. Figure 4 displays the system components and their relationships. Additionally, an application is shown to display the interfaces to the system.

The "raw" multimedia data streams are stored in external storage servers, not in the object-relational DBMS itself. This approach enables the use of specialized storage systems for multimedia objects capable of meeting the data delivery demands posed by presentations of continuous media objects. For example, an average compressed MPEG video requires the delivery of about 1.4 MB per second. We chose the Experimental Object Store (EOS) Storage Server from AT&T as an external storage manager because it can handle arbitrarily large objects effectively and provides random access to them [BP94].

The object-relational DBMS manages storage and retrieval of all metadata related to the multimedia objects. This is achieved by extending the DBMS with the Video Foundation DataBlade and CLF DataBlade. All discrete operations on the Continuous Long Fields that are not subject to time

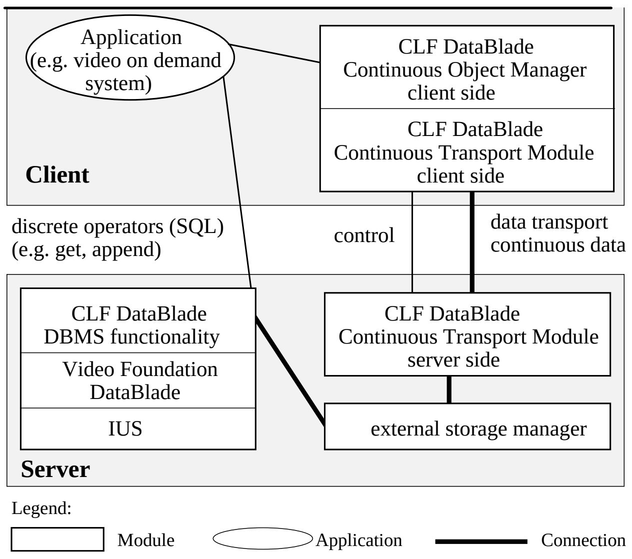


Fig. 4 : System Architecture

constraints, are performed by means of the DBMS functionality of the CLF DataBlade. For synchronous access operations, the object-relational DBMS grant direct access to the object stored on the external storage manager by the Continuous Object Management module on the client.

In addition to the supported DBMS functionality the CLF DataBlade offers continuous presentation support on the client side. This is implemented by the Continuous Object Management (COM) module on the client and the Continuous Transport Module. The Continuous Object Manager implements the client buffer management, described in section 2.2, and the adaptation techniques described in section 2.3.

The Transport Module connects the Continuous Object Manager with the external storage manager. It implements fast data access, fast delivery over the network, and transport of data requests from the client to the server and vice versa. The Transport Module sets up a continuous data connection to the external storage manager utilizing an appropriate network. The data connection and a corresponding control connection work in parallel to the other components of the DBMS and the application program. One data connection can consist of several logical channels to support composite presentations. Using this connection, time-dependent data are transferred from the server to client buffers. The control connection is responsible for transferring user commands as well as system control commands.

The application accesses a Continuous Long Field through the SQL interface of the Informix Universal Server. In this way all editing is performed on the objects. For presentation, the applications initialize the Continuous Object Management module based on the metadata of the object to be presented. The Transport module is initialized and the application is able to use the Continuous Object Management mechanism to request data, to set quality of service parameters or to communicate user interactions to the system.

3.4 THE CONTINUOUS LONG FIELD DATABLADE

DBMS functionality

Manipulation operations for multimedia objects provided by the Video Foundation DataBlade are on byte granularity and are comparable to simple file access mechanisms. The CLF DataBlade extends this functionality towards manipulation at the granularity of CODUs. Furthermore, it provides for the implementation of the virtual storage interface to the external storage manager EOS, and for the management of replicates of the same stream in different qualities.

For supporting editing and presentation, the basic metadata modelled in the Video Foundation DataBlade is extended by additional attributes. The integration of the additional metadata in the Video Foundation DataBlade database schema is illustrated in figure 5 .

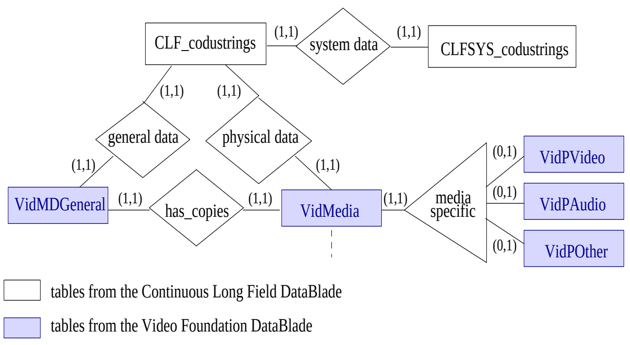


Fig. 5 : Database Schema of Video Foundation and CLF DataBlade

Two tables are defined as extensions to the schema of the Video Foundation DataBlade. The table CLF_codustrings holds the metadata the user has to supply to enable the system to handle, edit and present the continuous long field correctly. The table CLFSYS_codustrings holds information that is generated and maintained by the system, e.g., the entry number_of_codus that will be updated by any insert or append operation.

```
create table CLF_codustrings of type CODUSTRING_Type
( PRIMARY KEY (object_name),
  FOREIGN KEY (VideoID)
          (*references VidMDGeneral of the VF DataBlade*)
  FOREIGN KEY (MediaID)
          (*references VidMedia of the VF DataBlade*)
);

create table CLFSYS_codustrings of type
  CODUSTRING_SYSType
( FOREIGN KEY (object_name)
          (*references CLF_codustrings*)
);
```

The row types describing the tables are "CODUSTRING_Type" and "CODUSTRING_SYSType". They hold the metadata provided by the user and maintained by the system, respectively. The entries VideoID and MediaID are references to tables defined in the Video Foundation DataBlade. The row types are defined as follows.

```
create row type CODUSTRING_Type
( object_namevarchar (255),
        (*name of the object*)
  maximum_codu_size    integer,
        (*maximum size of the CODUs*)
  variable_codu_size  boolean,
        (*CODU size variable or not*)
  codus_per_second    integer,
        (*presentation speed in CODUs p/sec*)
  content_type        integer,
        (*type indicator: video, audio, other*)
  set_of_keywords     set (varchar (255) not null),
        (*keywords *)
  VideoID             integer,
  MediaID             integer
);

create row type CODUSTRING_SYSType
( object_name         varchar (255),
  number_of_codus     integer,
        (*number of CODUs in the object*)
  replicates          list (REPLICATE_Type not NULL),
        (*list of minor quality replicates*)
);
```

The datatype REPLICATE_Type serves as a representation of lower quality replicates of a Continuous Long Field. The CLF DataBlade defines two functions for the creation and deletion of Continuous Long Field instances. The function **create_codustring** generates a new Continuous Long Field in the database. The user has to provide metadata for the creation of an instance: the object's name, the maximum CODU size of the object, the variable CODU size indicator, the presentation speed and the media type indicator. Additionally, the user has to provide the identification of the external storage server consisting of a name and an identification number on which the object is to be stored. Create_codustring enters metadata provided by the user into the Video Foundation DataBlade and the CLF DataBlade schema tables and induces the creation of an initially empty external object. A sample call of create_codustring looks like this:

```
execute function create_codustring ('my_cs', 20, 'f', 1, 0,
'id_of_storage_server', 1);
```

The function **delete_codustring** removes a Continuous Long Field whose name is passed as a parameter from the database. It removes all entries in the metadata schema tables and also removes the external object. The SQL call of delete_codustring looks like this:

```
execute function delete_codustring ('my_cs');
```

The CLF DataBlade defines several functions to enable the editing of continuous objects. These functions operate on CODU indices. The editing functions adjust the relevant metadata (i.e. the number of CODUs and the size of the object) in the schema tables upon successful completion. The following functions are provided: append, insert, delete, insert_from_file, write_to_file, get. Additionally, the CLF DataBlade defines the function **make_replicate**, which is used to generate lower quality replicates of a Continuous Long Field, to be used for adaptation during a presentation. For illustration, we give one example of how the functions are applied. In the example it is assumed that a table "my_blobs" exists in the database which contains two data fields: name of type varchar (255) and b of type blob. This table holds an entry with the values ("first_blob", <blob data>).

The function **append** appends CODUs from a BLOB at the end of an object. Example use of append:

```
select append (cs, b)
from CLF_codustrings cs, my_blobs b
where cs.object_name ='first_cs' and b.name ='first_blob';
```

Continuous Presentation Support

Besides the refinements in the data management facilities described above, the main contribution of the CLF DataBlade is the integrated support for presentation and transport of continuous media. On the client side, this support is provided by means of a Continuous data API (CAPI). It supports continuous reading of continuous long field objects for presentation. It uses the Multimedia Data Direct Access Protocol (MDDA). The protocol provides methods for fetching the next COPU to be presented and for controlling the presentation.

The Continuous data API is implemented as a C++ library. Classes are provided to an application to initialize the Continuous Object Manager, to start a presentation, to request COPUs from the server, and to pass user interactions to the Continuous Object Manager. To initialize the Continuous Object Manager, an instance of "class **ContinuousObjectManager**" has to be created. A Continuous Long Field is represented by an instance of the "class **CODUString**". This class is initialized using the metadata stored in the table "CLF_codustrings". It offers the function "present" which initiates a presentation of the object. The "class **MDDA**" models the protocol interface for a presentation. It offers the function "getNextCOPU" which retrieves the next COPU in presentation direction. The function "dropNextCOPU" is used to inform the system that the first COPU fetched but undropped can be replaced in the client buffer. The protocol instances for an initialized presentation are retrieved from an instance of the "class **MDDA_Manager**" which the application has to create. The "class **QOS**" is used to specify Quality of Service parameters for a presentation. The instances of "class **COPU**" represent the COPUs that are fetched by using the functionality of "class MDDA". Figure 6 shows a simplified example of a presentation implemented using the described classes.

For the purpose of clarity and compactness, the example was stripped from all irrelevant code. The code fragment initializes the Continuous data API functionality by creating an instance of "class ContinuousObjectManager". The metadata for the continuous object to present (its object_name is "vid1") is fetched from the Informix Universal Server by means of the C++ interface of the Universal Server. An instance of class CODUString is created using the value of the "row type CODUSTRING_Type". A presentation with the id "video" is initialized, and the corresponding protocol instance is fetched from the MDDA_Manager. The presentation loop requests all COPUs in normal order. For each COPU, the pointer to the actual data is passed to the function "sentToDisplay" and, upon return from that function, the

```
void player ()
{
        // initialize CAPI functionality
new ContinuousObjectManager ();
        // initialize CODUString with metadata from
        // table CLF_codustrings
ITRow * row = query–>ExecOneRow (
        "select * from CLF_codustrings
        where object_name = 'vid1'");
CODUString * CODUString_to_present =
        new CODUString (conn, row);
CODUString_to_present–>present ("video", 1, 1);
        // initialize the COM
MDDA_Manager * mddaMan = new MDDA_Manager();
        // new instance of MDDA_Manager
MDDA * mdda = mddaMan–>connectToMDDA ("video");
        // get MDDA representation
While (play) {
        do {
                copu = theMDDA–>getNextCOPU ();
                        // request the COPU
                if (copu != NULL)
                        // COPU successfully retrieved
                *buffer = copu–>getAddr ();
        }
        sendToDisplay (buffer);
        // present COPU data
}}
```

Fig. 6 : A Presentation Example

COPU is dropped, meaning the system can replace it in the client buffer.

Annotation Support

The Video Foundation DataBlade provides a framework for a content based annotation of Continuous Long Fields. A content annotation for a continuous object always relates to a time interval on the object, e.g., a news video clip shows Mr. Clinton from second three through nine. Time intervals are modeled by the Video Foundation DataBlade by means of the datatype MedChunk which is composed of two precise locations on the timeline of the object. The time intervals can be described by one or more descriptions of four possible formats: text, image, audio or video. A described time interval is called stratum. It is stored in the table called "VidChunkMD" of the Video Foundation DataBlade. Several stratum values with a common property (e.g., video clips showing buildings) can be gathered in a relationship called "strata". Strata representations are inserted into the table called "VidStrata". A query can, for example, find all video clips that display buildings from a video archive. The result of this query would be a set of time intervals corresponding to videos and one or more descriptions for each time interval. Detailed description of the strata concept can be found in [SD92] and [DSP91].

The content-oriented metadata can be queried, but at the current state the presentation of query results (i.e., time intervals from Continuous Long Fields) is not supported. This is a future enhancement of the system.

4 RELATED WORK

Quite a number of research prototypes of multimedia database management systems have been developed over the last years, mostly on proprietary platforms. A nice overview, also with a classification of their most important features is given, for example, in [PS97]. From this overview one can take that most MM-DBMSs concentrate on storage and retrieval support, while presentation support is less frequent. One of the systems analyzed in this overview is the AMOS prototype that has been developed as an extensions of the object-oriented DBMS VODAK, and components of which were also used in the realization of the system described in this paper.

In [IKO96] a MM-DBMS prototype based on an object-oriented data model representing temporal, spatial, and agent models is described. Multimedia applications are modeled as a set of scripts which provide an interface to the user. Basically, a script consists of an identifier, temporal and spatial data, a set of streams, and QoS data. The real-time support for time-dependent data is done by an agent manager which negotiates the required QoS specifications with the system to satisfy the constraints and reserve the resources. Furthermore, the agent management checks deviations from expected values.

As an example of a commercial development the Oracle Universal Server is open for the development of so-called "cartridges" which are manageable objects. Furthermore, the language-neutral interface IDL (Interface Definition Language) of Corba allows the cartridge to identify itself to other objects in a distributed system. Cartridges can be programmed in multiple languages such as Java, C/C++, or SQL. The Oracle Video Server has been extended to manage new types of data, including video and audio, providing new ways to manage, manipulate, and deliver data in the networked economy [ORA96]. It will be seen in the future how the set of requirements for the integration of multimedia objects is solved.

5 CONCLUSION

In this paper we have shown in detail that supporting the particular requirements of a MM-DBMS, imposed in particular for the storage, delivery, and presentation of continuous media, can be integrated smoothly into today's standard extensible database management systems. We consider this an important step towards a simplified development of multimedia database applications.

We point out sample extensions of the basic mechanisms for continuous media handling of the current implementation that are the subject of our current work. They also illustrate the importance of using an open, extensible platform to deal with new requirements.

An extension that is required for adaptation with more complex media streams is the support for media-specific access operations for the preloader which are proposed in the CLF abstraction of section 2.1. One example are MPEG streams with dependent frames. These extensions will be provided within our framework as subtypes of the generic CLF data type and require additional metadata on the structure of the stream.

The quality adaptation mechanism is triggered within the local presentation context of the client. Thus, no global knowledge on resource consumption is considered. Such a mechanism needs to be realized on the database server. In [TKCWP97] a quality adaptation scheme for global quality adaptation and optimization is described that will be used to extend the mechanisms described in this paper. In addition, admission control mechanisms as described in [H97] will be realized to avoid an overload of the database server. Admission control techniques are used to guarantee a minimum quality for the presentation.

Future work will concentrate in particular on dealing with additional requirements and optimization possibilities when complexly composed multimedia presentations are involved and the consideration of application characteristics, e.g., the relevance of parts of a presentation, to support quality adaptations and optimizations.

REFERENCES

[AFK95] Apteker R. T., Fischer J. A., Kisimov V. S. and Neishlos H.: Video Acceptability and Frame Rate. In IEEE Multimedia, Fall 1995, pages 32-40.

[BP94] Biliris A., Panagos E.. EOS User's Guide. Release 2.2. AT&T Bell Laboratories, Murray Hill, NJ 07974, 1994.

[BAN95] Böhm K., Aberer K., Neuhold E. J.: Administering Structured Documents in Digital Libraries, Advances in Digital Libraries, Lecture Notes in Computer Science Vol. 916, Springer Verlag 1995.

[BKL96] Boll S., Klas W. and Löhr M.: Integrated Database Services for Multimedia Presentations. In S. M. Chung, Editor, Multimedia Information Storage and Management. Kluwer Academic Publishers, 1996.

[D95] Kotz-Dittrich A., Dittrich K. R.: Where Object-Oriented DBMSs Should Do Better: A Critique Based on Early Experiences. In Won Kim (ed): Modern DataBase Systems, ACM Press 1995.

[DHH93] Delgrossi L., Halstrick C., Hehmann D., Herrtwich R. G. , Krone O., Sandvoss J., Vogt C.: Media Scaling for Audiovisual Communication with the Heidelberg Transport System. In Proc. ACM Multimedia 1993, pages 99-104.

[DSP91] Davenport G., Smith T. G. A. and Pincever N.: Cinematic Primitives for Multimedia (Stratification Helps Story-Telling in Multimedia). In IEEE Computer Graphics and Applications, volume 11, number 4, July 1991, pages 64-74.

[HKR97] Hollfelder S., Kraiss A. and Rakow T. C.: A Client-Controlled Adaptation Framework for Multimedia Database Systems. In Proc. of the European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97), Darmstadt, Germany, Sept. 10-12, 1997.

[H97] Hollfelder S.: Admission Control for Multimedia Applications in Client-Pull Architectures. In Proc. of the Int. Workshop on Multimedia Information Systems (MIS), Como, Italy, September 1997.

[HL97] Hollfelder S. and Lee H-J.: Data Abstractions for Multimedia Database Systems. GMD Technical Report, No. 1075, Sankt Augustin, May 1997.

[IKO96] Ishikawa H., Kato K., Ono M., Yoshizawa N., Kubota K. and Kond A.: A Next-Generation Industry Multimedia Database System. In Proc. of Twelfth International Conference on Data Engineering, New Orleans, February/March 1996.

[MS96] Marcus S. and Subrahmanian V. S.: Towards a Theory of Multimedia Database Systems, In V.S. Subrahmanian, Sushil Jajodia, Editors, Multimedia Database Systems, Springer, 1996.

[MKK95] Moser F., Kraiss A. and Klas W.: L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS. In Proc. Int. Conf. of Very Large Data Bases 1995 (VLDB), Sept. 1995, pages 275-286.

[O96] Orji C.: Multimedia DBMS – Reality or Hype? In Nwosu, Thuraisingham, Berrs, Editors, Multimedia Database Systems, Kluwer, 1996.

[ORA96] Network Computing Architecture. An Oracle White Paper, September 1996.

[PS97] Pazandak P. and Srivastava J.: Evaluating Object DBMSs for Multimedia. In IEEE Multimedia Journal, vol 4, n. 3, Fall 1997.

[RKN96] Rakow T. C., Klas W. and Neuhold E. J.: Research on Multimedia Database Systems at GMD-IPSI. IEEE Multimedia Newsletter, Vol. 4, No 1, April 1996, pages 41-46.

[S96] Steinmetz A.: DiVidEd - A Distributed Video Production System. In VISUAL'96 Information Systems, VISUAL'96 Conference Proceedings, Melbourne 1996.

[SD92] Smith T. G. A., Davenport G.: The Stratification System: A Design Environment for Random Access Video. In Network and Operating Systems Support for Digital Audio and Video, Third International Workshop Proceedings, La Jolla, California USA, 12-13 November 1992.

[SH94] Steinmetz A., Hemmje M.: Konzeption eines digitalen Videoeditiersystems auf Basis des internationalen Standards ISO/IEE 11172 (MPEG-1). GMD-Studien Nr. 245, St. Augustin, Germany, 1994.

[TK96] Thimm H. and Klas W.: Delta-Sets for Optimized Reactive Adaptive Playout Management in Distributed Multimedia Database Systems. In Proc. of Int. Conference on Data Engineering, New Orleans, Louisiana, 1996, pages 584-592.

[TKCWP97] Thimm H., Klas W., Cowan C., Walpole J., Pu C.: Optimization of Adaptive Data-Flows for Competing Multimedia Presentational Database Sessions. In Proc. of IEEE Int. Conference on Multimedia Computing and Systems, June 3-6, 1997, Ottawa, Ontario, Canada.

[VFUG97] Video Foundation DataBlade Module User's Guide, Version 1.1, June 1997, INFORMIX Press, Menlo Park 1997.

[V95] VODAK V4.0 User Manual, GMD Technical Report No. 910, Sankt Augustin, April 1995.